Gartner Research

Publication Date: 21 December 2009 **ID Number: G00173105**

Composition and BPM Will Change the Game for Business System Design

Daryl C. Plummer, Janelle B. Hill

As enterprises continue to embrace collaborative development and business process management (BPM), IT, BPM and application development (AD) leaders will need to rediscover what it means to conduct "development" and fit that into a service-oriented architecture (SOA) world of complex systems with many interrelated parts. Business agility will depend on increased flexibility, adaptability and productivity, and the act of composition is one key way to enable this.

Key Findings

- Compositions — orchestrated assemblies of data, processes and services — will gradually replace applications as the primary incarnation of business software systems.
- Unlike traditional AD, which has been a programmer-driven effort, composition is a collaboration-driven effort that involves business subject matter experts, modelers, analysts, developers and architects.
- Model-driven approaches are becoming a primary method of system creation and maintenance. The model-driven approach is also a central theme of BPM, which promotes the idea that explicit process models are more effective than code models for continuous process improvement.
- The shift to composition will pose challenges in several areas, including cross-functional collaboration, life cycle management of software assets, process integrity and IT-business relationship management.
- Opportunities for organizations to source user interfaces (UIs), business and technical software services, and processes will expand as application vendors release software as a service (SaaS) and SOA-based versions of their products. Composition technologies will be packaged in new combinations, as integrated composition environments, to support various roles involved in creating and managing compositions.

Recommendations

- In the short term, foster a composition mind-set by instituting an educational change initiative to highlight differences between applications and compositions. Identify individuals with the appropriate skills and mind-set to fulfill composition roles (such as analyst, modeler, architect or component developer) that are appropriate to your organization and culture.

- In the medium term, identify tools needed to support the roles engaged in composition, establish life cycle management initiatives for services and compositions, and measure and track composition artifact reuse to measure effectiveness and productivity gains.
- In the long term, cultivate negotiation and relationship management skills, and institute intentional "modding" as a design principle.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

STRATEGIC PLANNING ASSUMPTIONS

Through 2014, the act of composition will be a stronger opportunity to deliver value from software than the act of development.

Through 2013, the top underrepresented skill in composition development will be relationship management.

By 2015, people will spend more than 80% of their time working collaboratively — but not necessarily face to face.

ANALYSIS

1.0 Introduction

Computing is undergoing a major evolution:

- Alternate models for acquiring and delivering systems — such as the Web for global deployment — have begun to emerge, forcing organizations to rethink how they fulfill their needs for those systems.
- BPM-enabling technologies are making business processes explicit, addressing the increased business requirement for operational transparency.
- Open source has illuminated the value of having large groups of people working independently to solve software problems in a community fashion.
- Web 2.0, SOA and cloud computing have broken up our systems into autonomous pieces, which can be stitched together to form solutions that break the constraints of what we formerly thought of as "applications."

Throughout all this, development must continue, but the act of development has been overshadowed by the act of deployment for so long that many organizations have yet to realize what's been lost in the evolution. As the movement toward composition, collaborative development and BPM continues, enterprise IT, BPM and AD leaders will need to rediscover what it means to conduct "development" and fit that into a world of cloud services, SOA and complex systems with many interrelated parts.

In this environment, increasing business agility will depend on higher-process flexibility and process-participant productivity. We believe the act of composition is a key way to enable this. In this report, we explore this subject by examining the following three areas:

- What the "act of composition" is, and how it will evolve during the next five years
- How composition, model-driven environments and SOA will change the game for software design
- How the act of composition will differentiate vendor strategies

2.0 What Composition Is and How It Will Evolve

2.1 Composition Defined

"Composition" has two related definitions. First, an individual "composition" is a solution that is *assembled* and *orchestrated* from independent parts:

- "Assembly" means bringing together independently built parts into a unified and "workable" whole.
- "Orchestration" means establishing a workflow through the assembled pieces to accomplish a goal.

Specifically: *Compositions are orchestrated assemblies of data, processes and services.*

A second, activity-based definition of "composition" — as in "the act of composition" (or the verb "compose") — describes the *process* of assembling and orchestrating part or all of a solution. The latter sense of the term is important, because the steps associated with building a composition demand a term that elevates this concept to a set of *actions* rather than a single *result*.

To be consistently effective at composition, the existing AD organization must first adjust its viewpoint of how work will be accomplished. Composition requires a broad influx of people into the process of development, who have independent responsibilities outside the realm of the application. Business analysts, graphical designers, service managers and even customers will participate in the creation of compositions, which eventually will replace applications as the primary incarnation of business software systems. Less technically oriented roles will participate well beyond the traditional "requirements definition" phase of the development life cycle and be actively involved in the assembly and orchestration efforts. More technically oriented roles will retain primary responsibility for creating the building blocks to be assembled. The movement from applications to compositions will require a shift in how we think about projects, organizations and collaboration across business and technical roles to achieve business goals. People will still use the word "applications," but they'll be living in a world of "compositions."

2.1.1 Compositions Versus Composite Applications

It is important to note that the term "composition" is not synonymous with the term "composite application" (see "Composite Applications Help Create Value by Blending SOA, Web 2.0, Cloud and Legacy Applications"). Composite applications are, in fact, a *type* of composition — in which the composed pieces are assemblies of interrelated application parts. However, there are many other types of compositions as well. For example:

- A composition may be a business process flow, where the assembled pieces are explicitly modeled processes running in a process server.
- Another type of composition is an orchestrated workflow, where the assembled pieces are tasks that are orchestrated by an orchestration engine.

2.2 The Act of Composition Versus the Act of Development

Composition is a collaboration-driven effort, involving modelers, analysts, developers, architects and subject matter experts. It often features:

- Modeled, explicit processes
- Flow control

- Iterative process refinement
- Iterative service evolution
- Variable design

Development, by contrast, has been a programmer-driven effort that involves mostly coders. It typically features:

- Coded, implicit processes
- Logic control
- "Waterfall-style" code refinement
- No service evolution
- Infrequent changes
- Mostly fixed design

Another key difference is that composition is *collaborative* and *change-centric,* while development has been historically been *deployment-centric*. Since the mid-1990s, IT organizations have focused intently on system deployment, and haven't expended nearly as much energy on the act of development.

Now, with technological and architectural advancements, the pendulum is swinging back toward building custom solutions rather than buying. However, the nature of the work involved in the act of development has changed in two key areas:

- There is more modularity and granularity in the component parts that can be incorporated into a system.
- It takes more collaboration to advance a system from its creation into any form required to deliver agility.

Collaboration, then, becomes a measure by which the act of composition will be evaluated. It's the mechanism that ensures that the resulting system and its requisite parts happen in the context of the required solution, not just in the context of one developer's view. In addition, collaboration helps expose the gaps between groups and capabilities that are intended to solve a given problem. It is the mechanism that brings together the viewpoints of business applications; process management; alternate delivery styles, such as cloud computing and software as a service (SaaS); Web development (for example, "mashups"); model-driven development; governance; service-oriented development of applications (SODA); and code-based development life cycles. Collaboration ensures that little is overlooked and nothing is left out of context.

2.3 Using a Model-Driven Approach to Composition

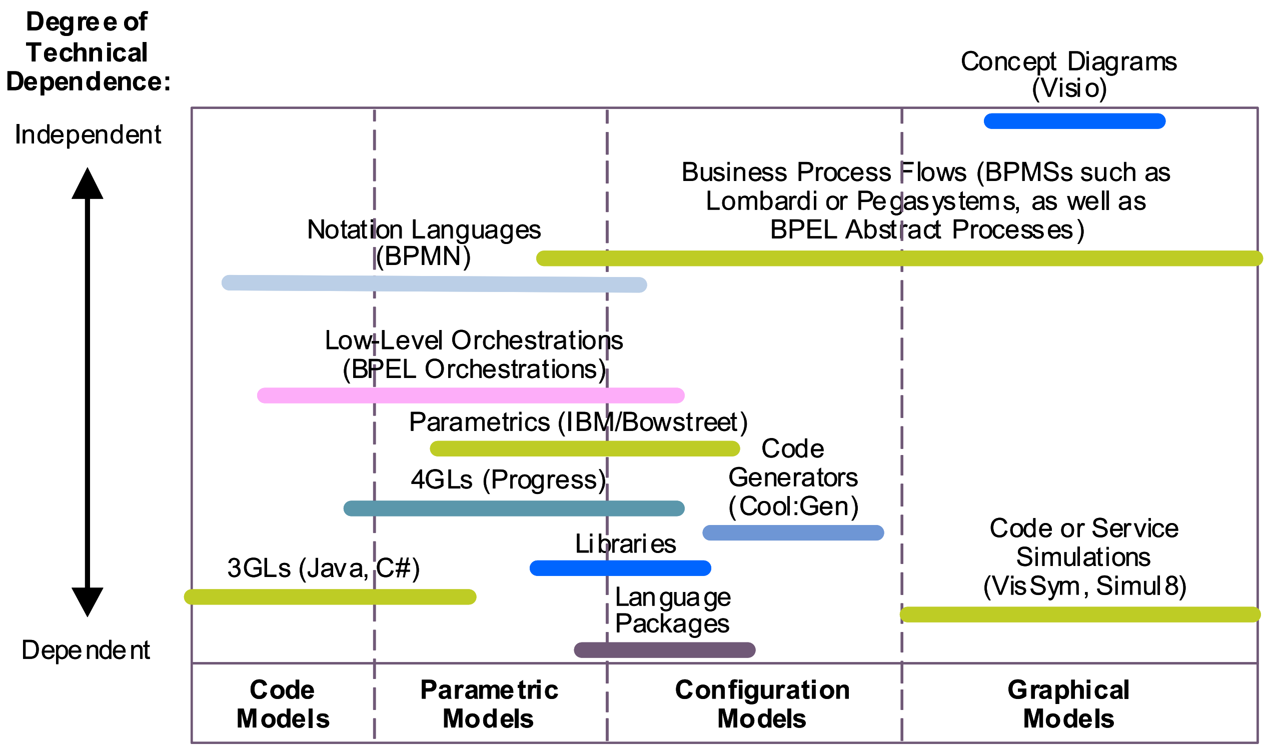
Models are an abstract representation of a concept or thing. The use of model-driven approaches to create and maintain applications and compositions is becoming a primary method of system creation. Model-driven applications and compositions are a central theme of BPM, which promotes the idea that explicit process models are more effective than code models for process improvement. The use of a model reduces the need for specialized programmers to manipulate code. This means that nontechnical people, such as business analysts, can participate in system design and creation. In addition, if the model is not "lost" at execution time, there is a much greater opportunity for monitoring the process, alerting and collecting key performance indicators

(KPIs) from a business perspective. Moreover, governance, deployment and life cycle management are greatly facilitated.

Beyond traditional code models, three types of models have emerged since the 1990s and become common: parametric, configuration and graphical-abstraction models (see "Three Types of Model-Driven Composition: What's Lost in Translation?"). All four models (see Figure 1) are described as follows:

- *Code models* describe the syntax and semantics of programming to hide the complexity of the underlying machine language. Examples include third-generation languages (3GLs).
- *Parametric models* are sets of related mathematical equations in which alternative scenarios are defined by changing the assumed values of a set of fixed coefficients (parameters).
- *Configuration models* use prebuilt packages, components, modules or libraries to enable a developer to configure a system from a preselected set of functions.
- *Graphical abstraction models* offer numerous advantages as the primary mode for model-driven systems. Textual representations underlie the graphical representations, but the generation of these lower-level "metamodels" can be automated.

Figure 1. Types of Model-Driven Composition



Degree of Technical Dependence:
Independent
Dependent
Concept Diagrams (Visio)
Business Process Flows (BPMSs such as Lombardi or Pegasystems, as well as BPEL Abstract Processes)
Notation Languages (BPMN)
Low-Level Orchestrations (BPEL Orchestrations)
Parametrics (IBM/Bowstreet)
Code Generators (Cool:Gen)
4GLs (Progress)
Libraries
Code or Service Simulations (VisSym, Simul8)
3GLs (Java, C#)
Language Packages
Code Models
Parametric Models
Configuration Models
Graphical Models

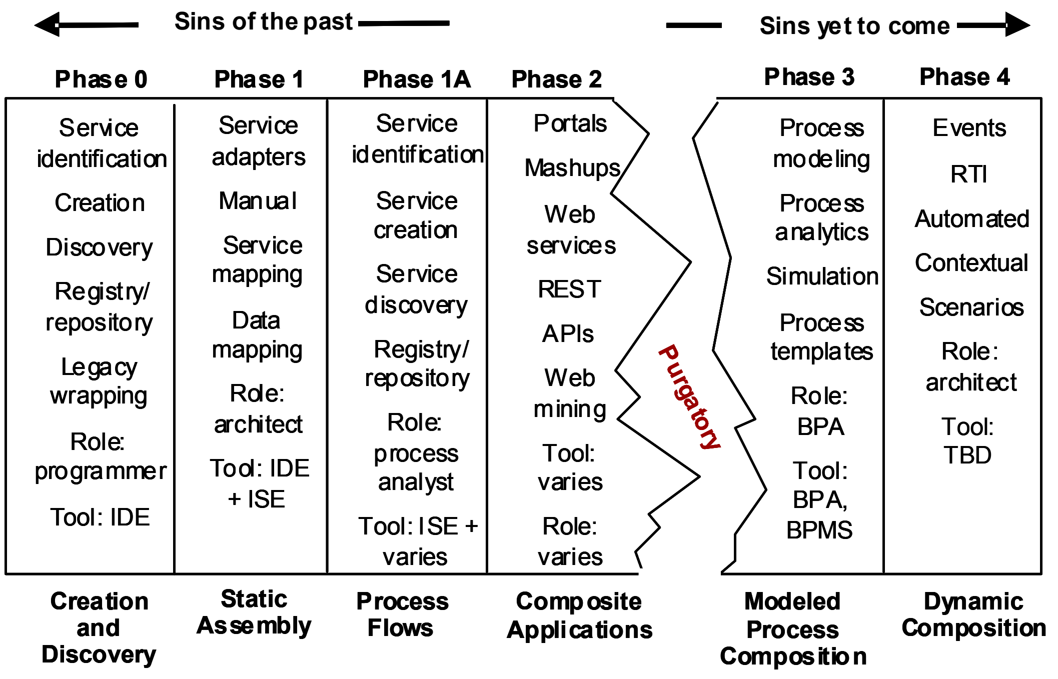
Source: Gartner (December 2009)

Model-driven applications and compositions are becoming commonplace. As business processes come to the fore, the need for nontechnical people to join the collaborative effort of building a system becomes paramount. Graphical abstraction models and their underlying textual models will enable organizations that seek to encourage this collaboration to accelerate the development and refinement of key business systems.

2.4 Gaining Composition Maturity

Organizations using composition will progress through phases of advancing maturity. These phases range from simple creation and discovery of the parts to be composed to dynamic composition, where the system selects the most-appropriate parts for building the system (see Figure 2).

Figure 2. Evolving Scenarios for Composition



Source: Gartner (December 2009)

Most organizations are just discovering how to create and find the necessary parts for composition. However, a larger difficulty looms: The movement from composing systems toward a more-modeled and dynamic style of system generation will require new skills and roles to be established. Service catalog managers and process analysts and architects will need to work together to determine how compositions will be best composed. Event managers will also be needed to ensure that the events used to develop dynamic processes are appropriate for any business context in which they must be used. Through 2013, the top underrepresented skill in composition development will be relationship management.

The future of composition will require many enterprises to cross a "chasm" or sorts before they truly understand how to generate consistent value. Crossing this chasm will require understanding shifts in the skills base, the organizational structure, the technologies being used, and the basis of collaboration in a world of compositions instead of on a team of developers.

3.0 How Composition, SOA and Model-Driven Approaches Will Change Software Design

3.1 Preparing for the Shift to New Development Models

The compositions of tomorrow will rely more on being model-driven than code-driven. This shift is only one of many changes related to the breakup of the application and the program into several features or characteristics that can be controlled outside the "container" running them. Transaction- and data-centric models are yielding to Web- and document-centric ones. This change will include a shift from legacy platforms focused on application execution to modern platforms focused on composition evolution. As Table 1 shows, characteristics of the modern platform include becoming more service-oriented and process-centric. In this respect, the styles of composition can move the focus of applications from deployment and execution back to construction.

Table 1. Modern AD Requires Modern Approaches

Characteristics of Legacy, Application Execution Platforms	**Characteristics of Modern Composition Evolution Platforms**
Transaction-centric	Web-centric
Code-driven	Model-driven
Legacy languages	Modern languages
Centralized	Distributed
IT-oriented	Service-oriented
Process-ancillary	Process-centric
Rigid	Agile
Autonomous	Federated
Enterprise-class	Global-class

Source: Gartner (December 2009)

One benefit is that systems will become more agile and flexible, rather than rigid and unchangeable. Ironically, systems that are built for change will in fact last longer than those that are built to last. This will help cope with the increasing pace of change demanded by business. However, this shift must be coordinated as a collaborative effort among the various participants in the development process. Governance will assume new significance, as the tracking, monitoring and enforcing of policies will be the linchpin of successful projects.

3.2 The Rise of "Modding" as a Design Principle

"Modding" is slang that refers to modifying a system beyond its original design parameters. We believe that it will be increasingly important for business processes, applications and services to be designed to accommodate "modding" — that is, they should be designed to be able to change, rather than remain static. Styles of composition such as BPM and SOA will be important enablers of this trend.

- Modding of strategic *process* designs will be enabled through the model-driven planning and execution capabilities offered by BPM suite (BPMS) products.
- Modding of strategic *functional* designs will be enabled through the "modular puzzle pieces" provided by SOA.

As digital natives grow into the workforce, the concept of modding (slang originally used to describe the alteration of PCs and gaming systems beyond their design parameters) will assume more significance. Intentional modding — that is, modification of a system beyond its original implementation, but within a framework for change — will become the norm for computing systems. Note that "modding" does not apply to the act of extending an existing application through wrapping or interfaces. The term refers to changing the function of the system itself, not just extending the system or adding functions around it. In the business world, modding will extend into a value proposition offered by the modification of business processes by business people in collaboration with the IT organization.

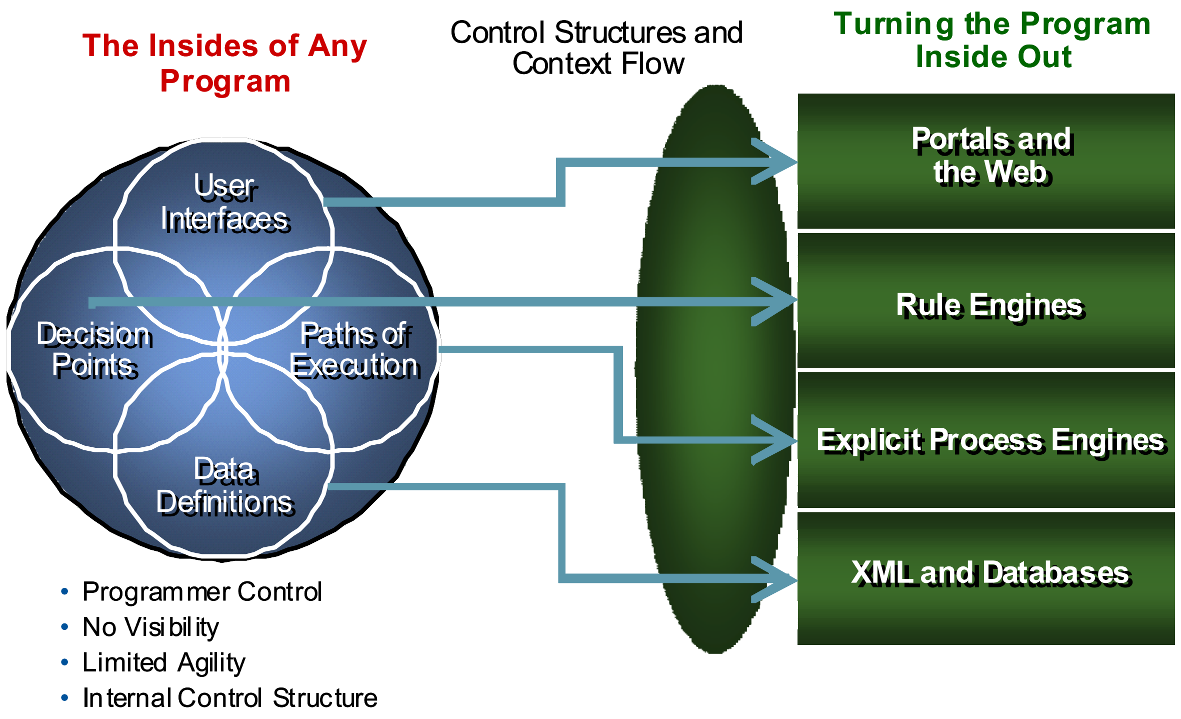
The key question is whether such modifications can be made while still maintaining process integrity and corporate standards. The answer is yes, but not without applying some managerial discipline to enable planned modification. SOA governance provides a technological framework for enabling this type of planning. Add to that the growth of BPM as a discipline, and the two areas form a complementary set of approaches for intentional business modding.

3.3 Exploding the Containers for Logic and Data

The "program" has been a useful construct throughout the history of computing. However, the need for agility has forced the traditional "software program" into a niche in which rapid change is not a dominant factor. Because agility is largely about change, the need for mechanisms to support change (as opposed to rigidity) is paramount.

Fortunately, modern computing architectures and tools are enabling IT departments to "crack open the egg" of the program to get to the "yolk" inside. Programs — which are typically composed of data, UIs, execution paths and decision points — are being crafted using composition/orchestration tools to provide a flexible framework for how a program functions, rather than the rigid containers of the past (see Figure 3).

Figure 3. Exploding the Program Leads to Agility and Dynamism



Source: Gartner (December 2009)

- *Portals and Web-based user experiences* have long since replaced single-function user interfaces as the primary means of delivering user access.
- *Rule engines* enable easy access to the decision points and routing logic in a system that may need to change from time to time.
- *Explicit process engines* enable multiple process flows to be formed to solve a given set of problems.
- *XML and dedicated database systems* have enabled data to move into a more free-form style.

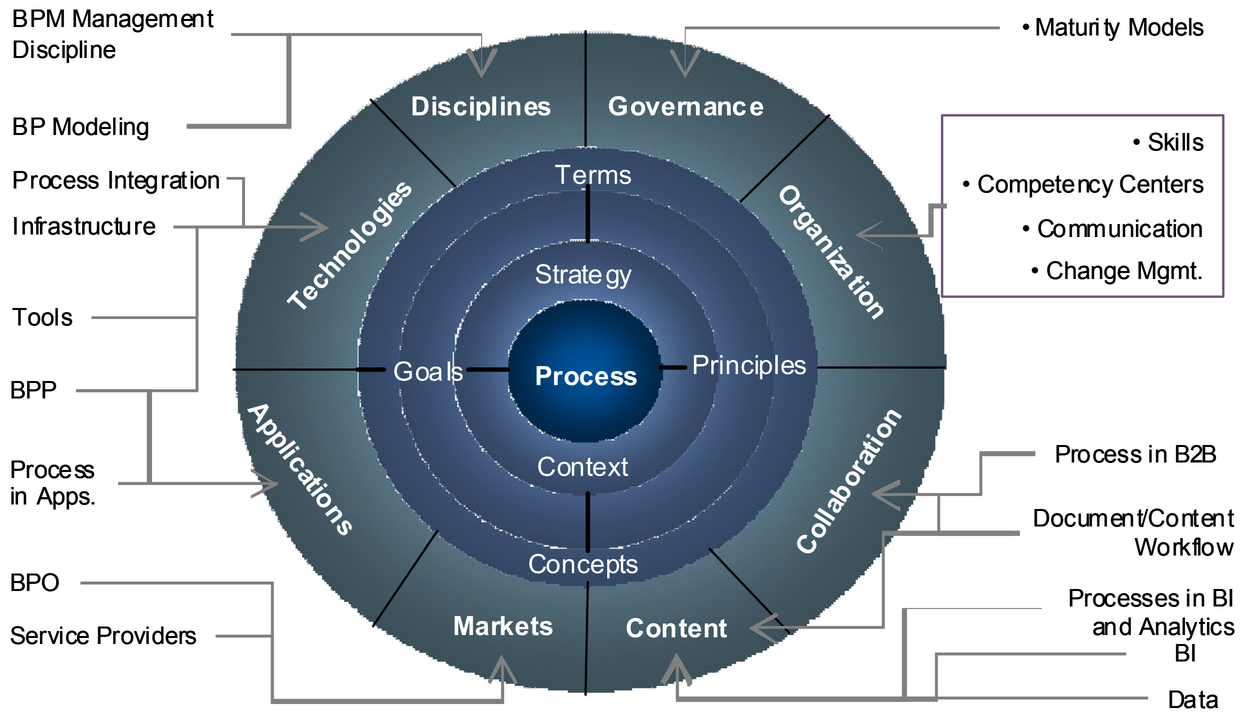
How does this relate to agility? The more options that are available for altering the way a system works without making it too fragile, the more likely it is that you'll be able to make the proper changes when necessary. Using BPMSs, much of this work can be automated, or at least made more accessible to the people who have domain expertise related to the system at hand.

To enable agility, organizations should remove as many barriers to change as possible, starting with barriers among containers for logic and data. SOA, XML and BPM combine to enable this transition.

3.4 Composition Affects Multiple Processes and Life Cycles

As compositions are created, the exposure of processes will be critical to understanding who must participate in the collaboration to sustain a solution. Multiple areas will be represented, and multiple life cycles and project styles must be understood. The process wheel shown in Figure 4 provides a view into how the different aspects of a process might be related. Business-to-business (B2B), business process outsourcing (BPO) and application processes are all related.

Figure 4. Process and Composition: Everyone Is Affected



Source: Gartner (December 2009)

For example, when a company outsources to a provider such as salesforce.com, data movement from the user company into salesforce.com's systems must be coordinated. This requires that some integration logic be created before the composition can be effectively delivered. If multiple companies are involved in the solution, the composition assumes a very collaborative need. How data is transformed and moved among the different parts of the system must be governed through some common understanding of architectural principles, goals, strategies and context.

Moreover, the multiple life cycles that must be managed will pose a larger challenge than that of traditional application life cycle management (ALM). Composition life cycle management differs from ALM in that it is layered across multiple life cycles for services, processes, data and applications. Metadata registries and repositories will be required to help manage all the moving parts throughout the life cycle.

In SOA and BPM projects, software services and explicit processes are designed and implemented. Services and processes have different life cycles compared with applications. Therefore, it is inappropriate to treat a composition of services and processes as if it were flowing through an application life cycle. Services evolve through a different cycle than applications, and can be split across different technologies and teams for quality assurance. Processes have life cycles that change, depending on the processes a business uses to evolve itself. Therefore, composition life cycle management is a multilayered effort.

3.5 Process Integrity Challenges and Solutions

Process integrity is the degree to which all process elements (such as tasks, participants and services) are intact and functioning in ways that ensure sustainability and long-term adaptation to changing conditions and human uses (see "Introducing Process Integrity: Critical to Business Applications, SOA Compositions and Processes"). Process integrity has three aspects:

- *Interaction integrity:* Users must be provided with up-to-date, secure access to information and content.
- *Transaction integrity:* Transactions must execute consistently and have the capability to recover as required.
- *Data/information integrity:* Data and other information must be complete, reliable and manageable.

Process integrity isn't guaranteed across distributed software services and composite applications. Therefore, the success of business applications and compositions depends heavily on a broad view of how much process integrity is required and how to deliver it when necessary.

Organizations that implement packaged business applications expect that, as users execute processes, the integrity of the interactions, transactions and data are built into the packaged offerings. In fact, protecting process integrity is one of the biggest justifications for acquiring an integrated business application suite instead of a set of best-of-breed applications. With new, alternative application consumption and creation models; however, process integrity isn't guaranteed, so users that adopt SaaS, SOA, model-driven composition and next-generation service-oriented business application (SOBA) projects must view process integrity in a new way. Distributed systems and distributed companies have increased the need for distributed transactions and interactions, which have different process integrity requirements.

Compositions are built, deployed and maintained across different working groups, companies and even application/composition vendors. This is changing the landscape of process integrity so much that inherent process integrity is no longer guaranteed. Therefore, organizations making compositions must make process integrity a priority. They must focus more effort on ensuring interaction integrity rather than just distributed transactions, and adopt a discipline associated with elevating human interaction in the guarantee of process integrity.

3.6 The Relationship Management Challenge

The movement from applications to compositions will require a shift in how we think about projects, organizations and collaboration to achieve the goal. Old-style application delivery approaches don't fit well with new SOA and BPM compositions. Affected areas include:

- **Costing and counting:** How does IT count or price distributed and shared resources, such as services? This will move from function points to "service points."
- **Testing and quality assurance:** Who tests what, and at what level? This will move from a centralized process to a multilevel one.
- **Project management:** As discrete projects become less common and "interleaved ones" are more the norm, project management will occur across multiple project levels.
- **Portfolio management:** An SOA portfolio is different from an application portfolio, and includes people and procedures. Therefore, portfolio management will become less asset-based and more governance-based.

- **Roles and organization:** The user of vendor products is no longer just a programmer and an architect. Businesspeople are now part of the buying decision.
- **Service provider relationships:** Today, these relationships are based on a "waterfall" application life cycle construct, but they will need to become more agile, iterative and collaborative.

An organization seeking to manage compositions must manage distributed relationships and agreements for collaborative teams. Fewer than 2% of IT personnel are trained in relationship management, and this must change. During the next five years, relationship management will be as important as project management in composition projects. By 2015, people will spend more than 80% of their time working collaboratively — but not necessarily face to face.

4.0 How Composition Will Differentiate Vendor Strategies

4.1 Vendors Will Respond to New Software Consumption Models

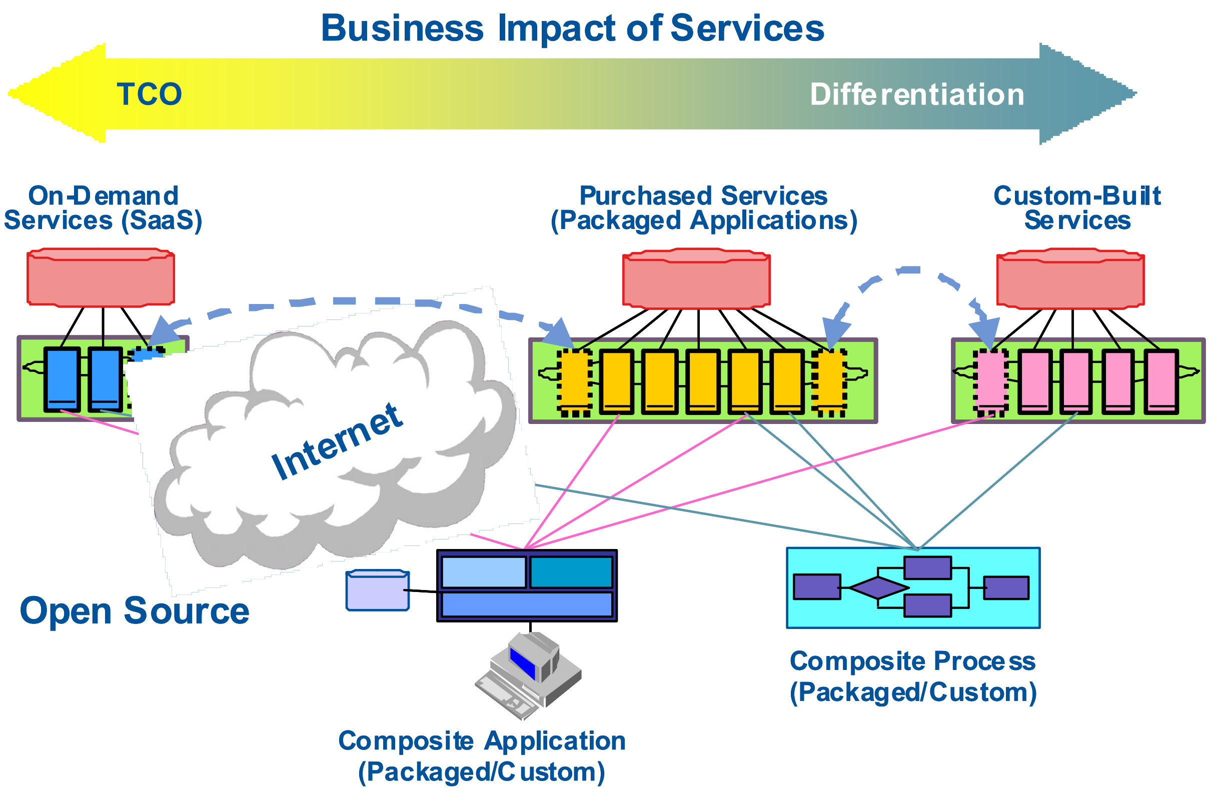
Most organizations that have adopted SOA have implemented their services by wrapping pre-SOA applications or developing new, native services. These are mostly "consumed" by custom-developed composite applications or processes. To implement these applications, organizations increasingly leverage services that are sourced "on demand" through cloud or SaaS models as providers of these solutions, such as how salesforce.com or Workday, open their applications to service-oriented interfaces.

During the next five years, opportunities for users to source services as well as composite applications and processes will expand as packaged application vendors release SOA-based versions of their products, and deploy SaaS offerings based on these new products. The traditional "buy versus build" tension will become even more acute for users, because the choices will become more granular. A third option, on-demand, will become increasingly available and viable.

Hence, with three main options — on-demand, purchased and custom-built services (see Figure 5) — the choice will become a "build versus buy versus on-demand" one:

- Organizations will likely turn to the on-demand option for sourcing the more-standardized and nondifferentiating services, with the goal of minimizing costs.
- They will pursue the buy option to support standardized services — and, increasingly, packaged composite applications (PCAs) and packaged integrating processes (PIPs) — that require some level of customization, or that can't be "leased" via on-demand options for security, privacy or other reasons.
- A certain number of services and composite applications/processes will continue to buy custom-built when it comes to supporting the most-differentiating, nonstandardizable business processes.

Figure 5. Choices Will Multiply With Alternate Software Consumption Models


Business Impact of Services
TCO
Differentiation
On-Demand Services (SaaS)
Purchased Services (Packaged Applications)
Custom-Built Services
Internet
Open Source
Composite Process (Packaged/Custom)
Composite Application (Packaged/Custom)

Source: Gartner (December 2009)

Overall, to minimize costs and maximize business differentiation, organizations should adopt a mixed build/buy/on-demand approach to implementing services in their SOA adoption strategies.

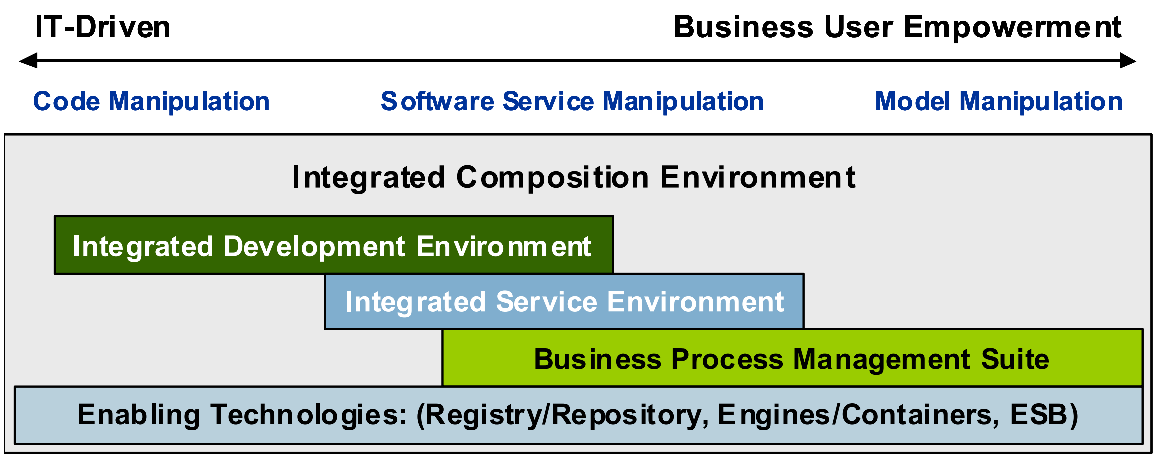
4.2 Integrated Composition Environments

An integrated composition environment (ICE) is the set of technologies necessary to support the act of composition across the life cycles of multiple artifacts. These artifacts include services, processes, components, objects and the resulting compositions that are used to create. ICEs require design and deployment information for business processes, software services, policies and methods.

ICEs include three subtypes (see Figure 6):

- Integrated development environments (IDEs) enable developers to take a code-centric approach to the application life cycle.
- Integrated service environments (ISEs) enable developers and architects to take a service composition approach to the application life cycle.
- BPMSs enable IT professionals and business roles to manipulate applications throughout their life cycles via a model-driven approach. BPMS vendors are evolving their products in the context of the ICE. By 2010, BPMS products will be the leading form of ICE used for creating a business process platform (BPP) model.

Figure 6. Three Types of IDEs

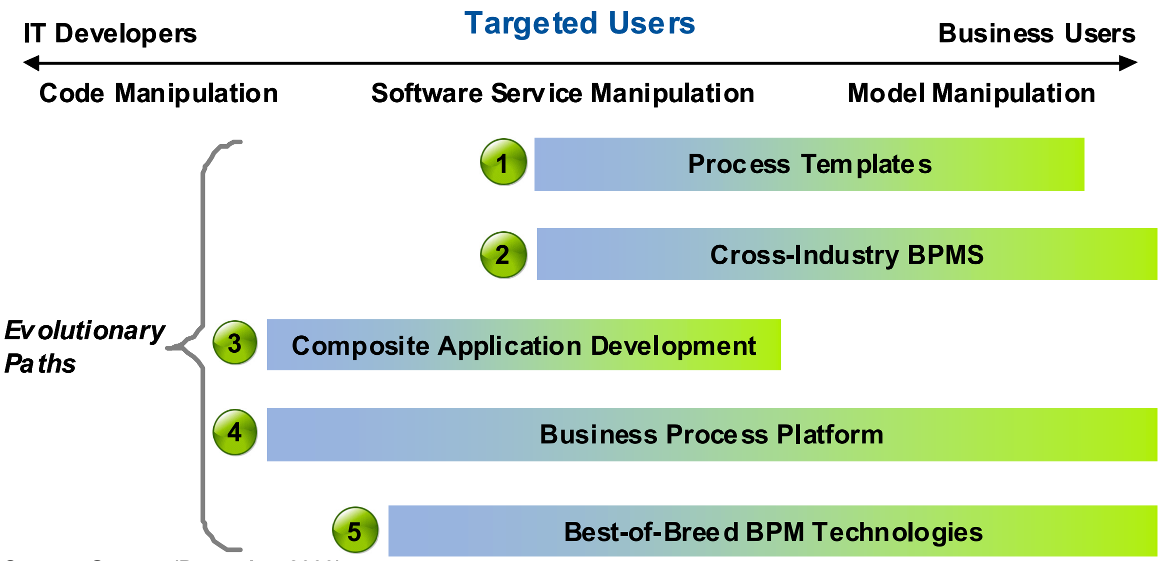


IT-Driven
Business User Empowerment
Code Manipulation
Software Service Manipulation
Model Manipulation
Integrated Composition Environment
Integrated Development Environment
Integrated Service Environment
Business Process Management Suite
Enabling Technologies: (Registry/Repository, Engines/Containers, ESB)

Source: Gartner (December 2009)

A BPMS is a model-driven ICE that empowers the business users' participation throughout the process improvement life cycle (see "Three Types of Model-Driven Composition: What's Lost in Translation?" and "BPMS: Today's Best ICE Choice for Your BPP"). We believe BPMS vendors will follow one of the five evolutionary paths depicted in Figure 7, which places each of the five along the continuum from code manipulation (which is the purview of developers) to model-driven manipulation of processes (which is more suitable for those with business-oriented skills). For more information, see "Signs That a BPMS Vendor Is Following One or More Technology Evolutionary Paths."

Figure 7. Five Evolutionary Paths of BPMS Vendors

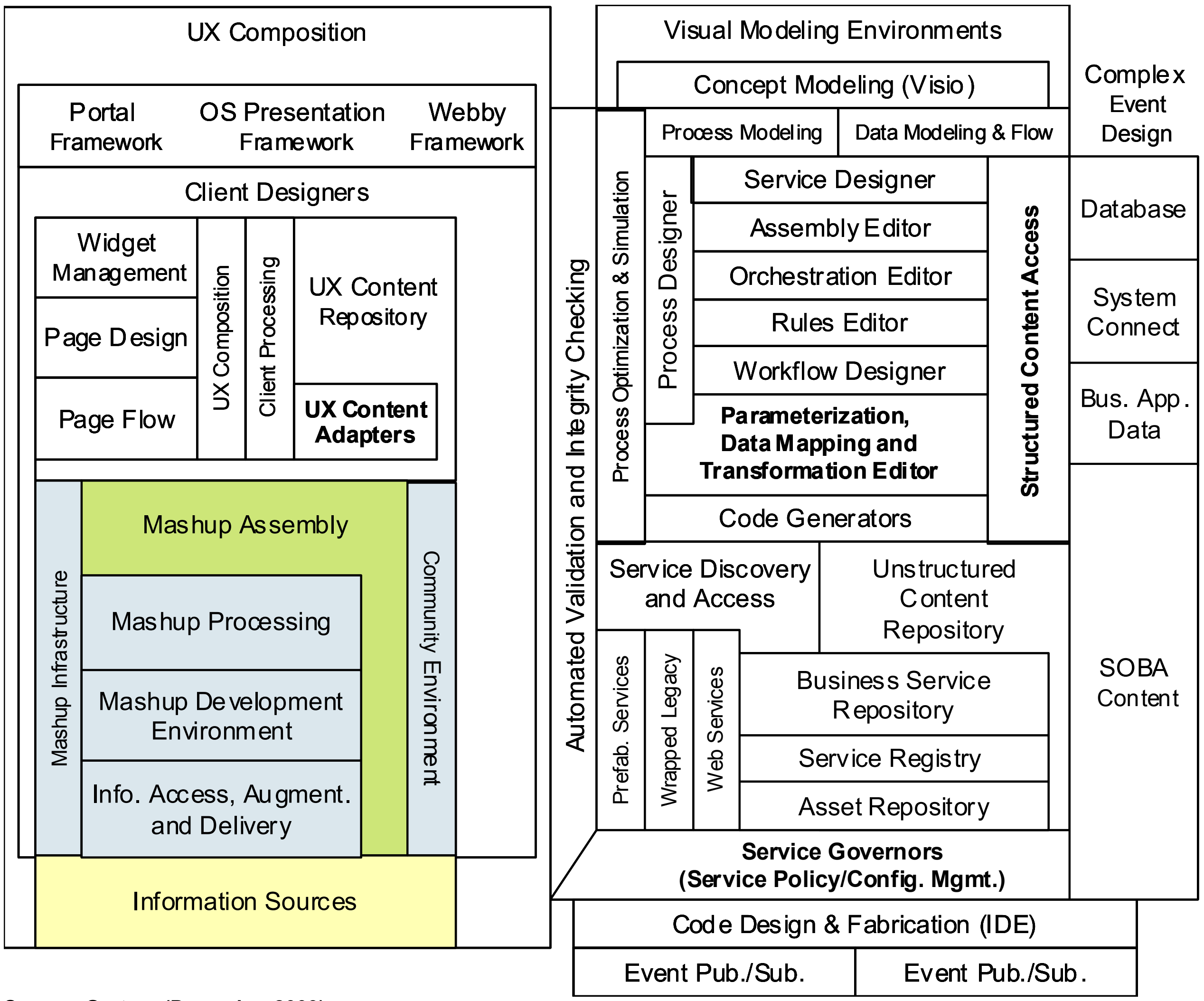


Source: Gartner (December 2009)

4.3 The ICE Reference Architecture

The act of composition inevitably will lead to a common set of capabilities associated with the products used to create the compositions. The reference architecture shown in Figure 8 attempts to bring together many of the capabilities essential to a complete composition environment. Many runtime elements are associated with this environment. It isn't just a design environment, and must be coupled with the entire life cycle.

Figure 8. Reference Architecture for an Integrated Composition Environment



Source: Gartner (December 2009)

Creating a composite application by writing code to assemble objects or components can lead to complexity and inconsistent results. However, modeling and assembling process compositions from components in an SOA is proving to be more efficient and effective than writing code. Many tools traditionally used by professional IT developers have been extended into complete environments supporting this composition and assembly design/development paradigm.

Combining application infrastructure (such as orchestration engines, logic containers, service registries and service buses) with these design and composition environments is described as "integrated composition technologies" in the BPP model. Integrated composition technologies require design and deployment information for business processes, software services, policies and methods. For example, the processes may have been defined in a business process analysis tool, with policies in a policy management tool and rules in a business rule engine. The design effort may have used specific business process methodologies and project management tasks to develop the process and identify potential business services to build or reuse.

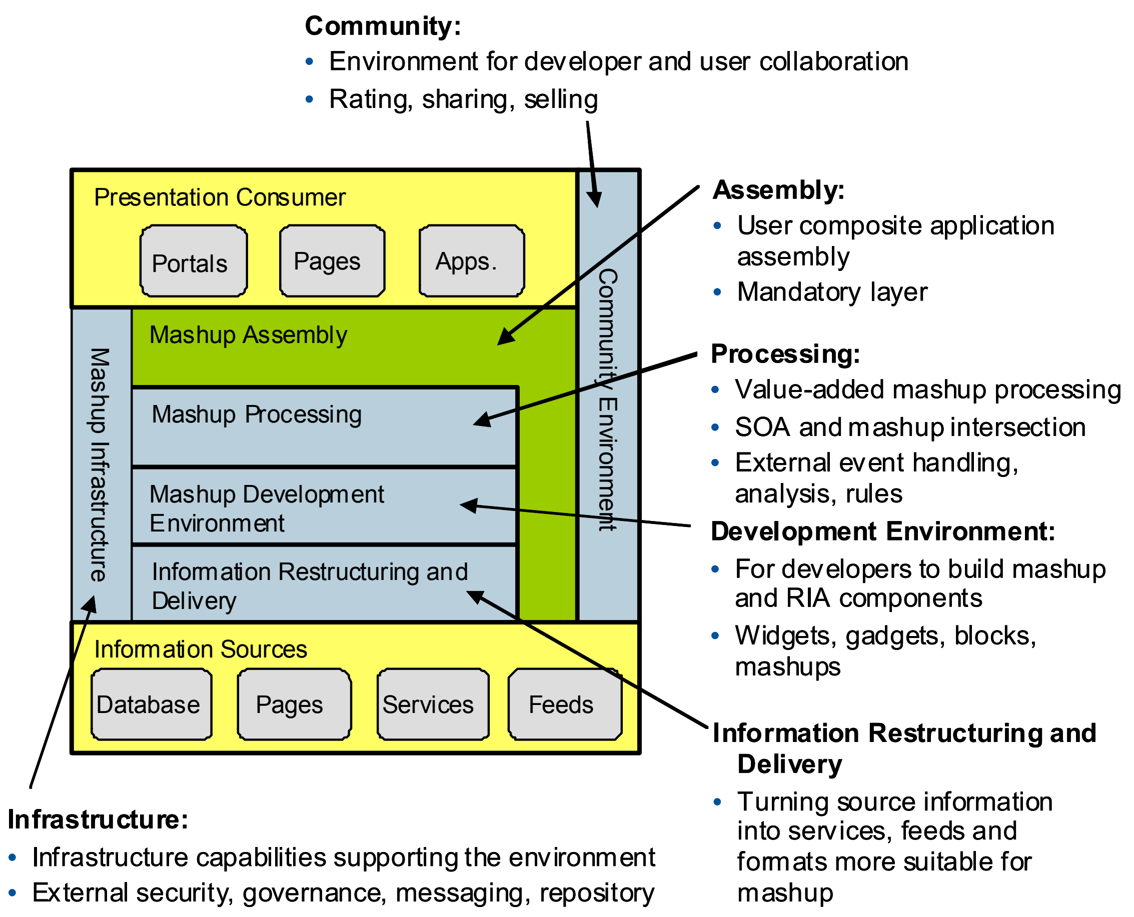
4.4 Enterprise "Mashups": The Low-Hanging Fruit of Composition

Gartner believes that, through at least 2010, "mashups" will be the dominant style of opportunistic composite applications. Enterprises are investigating how to develop mashups from a Web hobby into enterprise-class systems to augment their models for delivering and managing applications. Established and startup vendors are entering the enterprise mashup market with promising tools. Application architects must investigate this growing space for the significant and transformational potential it may offer their enterprises.

An enterprise mashup reference architecture is emerging as enterprises assemble systems and vendors craft offerings. The Gartner mashup reference architecture (see Figure 9) consists of six layers residing between the information sources and presentation consumers:

- Mashup assembly
- Mashup processing
- Mashup development environment
- Information restructuring and delivery
- Mashup infrastructure
- Mashup community environment

Figure 9. Mashup Reference Architecture


Community:
Environment for developer and user collaboration
Rating, sharing, selling
Presentation Consumer
Portals
Pages
Apps.
Community Environment
Assembly:
User composite application assembly
Mandatory layer
Mashup Assembly
Mashup Infrastructure
Processing:
Value-added mashup processing
SOA and mashup intersection
External event handling, analysis, rules
Mashup Processing
Mashup Development Environment
Development Environment:
For developers to build mashup and RIA components
Widgets, gadgets, blocks, mashups
Information Restructuring and Delivery
Information Sources
Database
Pages
Services
Feeds
Information Restructuring and Delivery
Turning source information into services, feeds and formats more suitable for mashup
Infrastructure:
Infrastructure capabilities supporting the environment
External security, governance, messaging, repository

Source: Gartner (December 2009)

The mashup assembly layer is mandatory for enterprise mashup tools. The remaining layers are optional, depending on specific requirements. Vendors offering layer capabilities, but not a mashup assembly capability, are more accurately described as "mashup enablers" rather than mashup tool providers. Application architects should use the enterprise mashup reference architecture to examine the evolving mashup technology space.

5.0 Conclusion and Recommendations

Gartner expects compositions to gradually replace traditional applications as the primary incarnation of business software systems. In addition to traditional developers, a number of other parties will collaborate in the act of composition, including business analysts, service managers and even customers. As the movement toward composition, collaborative development and BPM continues, enterprise IT, BPM and AD leaders will need to rediscover what it means to create business systems in an SOA world with many interrelated parts. Our recommendations for programmers, architects, IT managers and business analysts include the following:

In the short term (i.e., during the coming quarter):

- Foster a composition mind-set by instituting an educational change initiative to highlight differences between applications and compositions.
- Begin to identify individuals with the appropriate skills and mind-set to fulfill composition roles (such as analyst, modeler, architect or component developer) that are appropriate to your organization and culture.

In the medium term (during the coming year):

- Identify the tools you'll need to support the various roles engaged in the act of composition.
- Establish a service life cycle and a composition life cycle management initiative.
- Track composition artifact reuse to measure effectiveness and productivity gains.

In the long term (next year and beyond):

- Cultivate negotiation skills and relationship managers.
- Institute intentional "modding" as a design principle.

REGIONAL HEADQUARTERS

Corporate Headquarters
56 Top Gallant Road
Stamford, CT 06902-7700
U.S.A.
+1 203 964 0096

European Headquarters
Tamesis
The Glanty
Egham
Surrey, TW20 9AW
UNITED KINGDOM
+44 1784 431611

Asia/Pacific Headquarters
Gartner Australasia Pty. Ltd.
Level 9, 141 Walker Street
North Sydney
New South Wales 2060
AUSTRALIA
+61 2 9459 4600

Japan Headquarters
Gartner Japan Ltd.
Aobadai Hills, 6F
7-7, Aobadai, 4-chome
Meguro-ku, Tokyo 153-0042
JAPAN
+81 3 3481 3670

Latin America Headquarters
Gartner do Brazil
Av. das Nações Unidas, 12551
9° andar—World Trade Center
04578-903—São Paulo SP
BRAZIL
+55 11 3443 1509